M450

Testdoubles

Ein Testdouble ist ein Latzhalter für eine Komponente, welche nicht vorhanden ist oder nur schlecht getestet werden kann.
Vorteile:

- Auch ohne konkrete Komponenten kann unsere Klasse getestet werden
- Der Test der Klasse ist isoliert von der Implementierung der gedoubelten Komponente
- Der Test ist jederzeit reproduzierbar, also nicht vom aktuellen Verhalten des Services abhängig

Komponententest

- **Bedeutung:** Der Komponententest ist die erste Teststufe im Softwareentwicklungslebenszyklus. Hier werden einzelne Softwarekomponenten (z.B., Module, Klassen) isoliert getestet, um sicherzustellen, dass sie gemäß ihren Spezifikationen funktionieren.
- **Voraussetzungen:** Die Komponenten müssen bereits entwickelt sein, bevor der Test durchgeführt wird. Es können verschiedene Teststrategien wie White-Box-Tests oder Black-Box-Tests angewendet werden.
- **Abgrenzung:** Der Fokus liegt auf der isolierten Überprüfung der Funktionalität einer einzelnen Komponente. Externe Abhängigkeiten werden durch Platzhalter oder Testtreiber ersetzt.
- **Zusammenhänge:** Die Ergebnisse des Komponententests dienen als Grundlage für den Integrationstest, um sicherzustellen, dass die Komponenten miteinander interagieren.

Integrationstest

- **Bedeutung:** Der Integrationstest überprüft die korrekte Zusammenarbeit von miteinander verbundenen Komponenten. Dies kann nach verschiedenen Integrationsstrategien erfolgen, wie Top-down-Integration, Bottom-up-Integration, Ad-hoc-Integration oder Backbone-Integration.
- **Voraussetzungen:** Die zu integrierenden Komponenten müssen einzeln erfolgreich getestet worden sein. Es kann eine schrittweise Integration erfolgen, bei der höhere Schichten schrittweise mit untergeordneten Komponenten integriert werden.
- **Abgrenzung:** Der Fokus liegt auf der Interaktion zwischen den Komponenten. Es werden Platzhalter oder Testtreiber verwendet, um noch nicht integrierte Komponenten zu simulieren.
- **Zusammenhänge:** Die Ergebnisse des Integrationstests dienen als Grundlage für den Systemtest, um sicherzustellen, dass das Gesamtsystem korrekt funktioniert.

Systemtest

- **Bedeutung:** Der Systemtest prüft das integrierte Gesamtsystem, um sicherzustellen, dass es die spezifizierten Anforderungen erfüllt. Es wird aus der Perspektive des Kunden und Endanwenders betrachtet.
- **Voraussetzungen:** Der Integrationstest muss erfolgreich abgeschlossen sein. Die Testumgebung sollte der späteren Produktivumgebung so nahe wie möglich kommen.
- **Abgrenzung:** Der Fokus liegt auf der Gesamtfunktionalität und -performance des Systems. Es werden Tests auf Akzeptanz, Benutzerfreundlichkeit und Systemkonfiguration durchgeführt.
- **Zusammenhänge:** Die Ergebnisse des Systemtests dienen als Grundlage für den Abnahmetest.

Abnahmetest

- **Bedeutung:** Der Abnahmetest erfolgt aus der Sicht des Kunden oder Endanwenders. Es überprüft, ob das gelieferte System die vertraglichen und geschäftlichen Anforderungen erfüllt.
- **Voraussetzungen:** Das System muss erfolgreich den Systemtest bestanden haben. Die Testumgebung entspricht der Produktivumgebung des Kunden.
- **Abgrenzung:** Der Fokus liegt auf der Akzeptanz und Zufriedenheit des Kunden mit dem gelieferten System.
- **Zusammenhänge:** Der Abnahmetest ist die letzte Teststufe vor der Auslieferung an den Kunden und stellt sicher, dass die Software den Erwartungen des Kunden entspricht.

Dynamische Tests

Tests die während der Ausführungs des Programs durchgeführt werden (MSTest Projekt). Z.B. Unit-Tests, Integrationstests.

Vorteile:

- Dynamische Tests spiegeln das tatsächliche Verhalten der Software
- Dynamische Tests decken Fehler die nur in der Laufzeit auftreten sehr gut auf.

Nachteile:

- Es können nie alle Ausführungsszenarien getestet werden
- Tests müssen ausprogrammiert und ständig angepasst werden

Staatische Tests

Tests, die ohne Ausführung des Programms durchgeführt werden. Z.B. Code-Analyse Tools, Code-Review

Vorteile:

- Frühzeitige Erkennung von Fehlern
- Statische Analysewerkzeuge können schnell große Codebasen durchsuchen

Nachteile:

- Gefahr von Fehlalarm
- Statische Analysen können nicht alle Laufzeitaspekte und tatsächlichen Verhaltensweisen einer Anwendung berücksichtigen

Top-down-Integration

- Start mit oberster Komponente, die selbst nicht aufgerufen wird
- Platzhalter:
  - Untergeordnete Komponente wird mit Platzhalter ersetzt
- Testtreiber
  - Bereits getestete übergeordnete Komponenten

Vorteil
- Es werden keine Testtreiber benötigt
- Vertikale durchgängiger Proof of Concept (Risiko reduzieren)
- Frühes Feedback von Kunde möglich

Nachteil
- Noch nicht integrierte Komponenten müssen durch Platzhalter ersetzt werden

Bottom-up-Integration

- Start mit tiefster Komponente
- Schrittweise Komponenten der nächst höheren Schicht zufügen

Vorteil
- Keine Platzhalter benötigt
- Einfach über Unittests machbar

Nachteil
- Testtreiber benötigt
- Spätes Feedback von Kunde

Teststufen

Strategien für Integration und Integrationstest

Abnahmetest-Strategien

Ad-hoc-Integration

- In Reihenfolge, wie Komponenten verfügbar werden
- Vorteile
  - Frühes Testen, sobald Komponenten verfügbar
- Nachteil
  - Testtreiber notwendig
  - Platzhalter notwendig

Big-Bang-Integration

- Nicht inkrementelle Integration aller Komponenten, erst am Schluss wenn alle verfügbar sind
- Vorteil
  - ?
- Nachteil
  - Lange Wartezeit bis alle Komponenten verfügbar sind ist verlorene Testzeit
  - Fehlerwirkungen treten geballt auf, sehr schwierig, System überhaupt zum Laufen zu bringen.

Backbone-Integration

- Programmskelett, in das Komponenten schrittweise eingehängt werden können
- Cotinuous Integration CI
- Vorteil
  - Komponenten können in beliebiger Reihenfolge integriert werden durch standardisierte Schnittstellen
- Nachteil
  - Backbone-Umgebung kann recht aufwendig sein

Test auf vertragliche Akzeptanz

- **Erläuterung:** Bei der Entwicklung von Individualsoftware führt der Kunde in Zusammenarbeit mit dem Lieferanten einen vertraglichen Abnahmetest durch. Dieser Test basiert auf den im Entwicklungsvertrag festgelegten Abnahmekriterien. Der Kunde entscheidet auf Grundlage der Testergebnisse, ob das Softwaresystem mangelfrei ist und die vertraglich geschuldete Leistung erfüllt.
- **Testkriterien:** Die im Entwicklungsvertrag festgeschriebenen Abnahmekriterien und ggf. relevante gesetzliche Vorschriften, Normen oder Sicherheitsrichtlinien.

Test der Benutzerakzeptanz (Benutzerabnahmetest)

- **Erläuterung:** Der Test der Benutzerakzeptanz erfolgt, um sicherzustellen, dass verschiedene Benutzergruppen das System akzeptieren und effektiv nutzen können. Dieser Test wird vom Kunden organisiert, der auch die Testfälle auswählt, basierend auf Geschäftsprozessen und typischen Anwendungsszenarien.
- **Testkriterien:** Unterschiedliche Erwartungen der Anwendergruppen an das System und deren Zufriedenheit.

Akzeptanz durch den Systembetreiber

- **Erläuterung:** Dieser Test zielt darauf ab sicherzustellen, dass das neue System nahtlos in die vorhandene IT-Landschaft des Systembetreibers integriert werden kann. Aspekte wie Backup-Routinen, Wiederanlauf nach einer Systemabschaltung, Benutzerverwaltung und Sicherheitsaspekte werden überprüft.
- **Testkriterien:** Integration des Systems in die bestehende IT-Landschaft, Backup-Routinen, Wiederanlaufprozesse, Benutzerverwaltung und Sicherheitsaspekte.

Feldtest (Alpha- und Beta-Tests)

- **Erläuterung:** Wenn die Software in verschiedenen Produktivumgebungen betrieben werden soll, wird ein Feldtest durchgeführt. Dies kann in Form von Alpha- und Beta-Tests geschehen. Alpha-Tests finden beim Hersteller statt, während Beta-Tests beim Kunden durchgeführt werden.
- **Testkriterien:** Identifizierung von Einflüssen aus verschiedenen Produktivumgebungen, Anpassung der Software entsprechend den Rückmeldungen der Kunden.

Abnahmetest in agilen Projekten

- **Erläuterung:** In agilen Projekten ist das Ziel, Feedback frühzeitig einzuholen. Akzeptanztests erfolgen iterativ, wobei der Fokus auf dem Feedback zu Verbesserungen für das nächste Release liegt.
- **Testkriterien:** Erfahrung der Funktionalität durch den Kunden, kontinuierliches Feedback für iterative Verbesserungen.

Sie erläutern den Zusammenhang bzw. die Unterschiede zwischen Test und Produktivumgebung

Die Testumgebung dient der Entwicklung und Überprüfung von Software, ohne Auswirkungen auf den produktiven Betrieb. Hier werden neue Funktionen getestet, Fehler behoben und Leistung optimiert. Die Produktivumgebung ist hingegen die tatsächliche, in Echtzeit laufende Umgebung, in der die Software von Endbenutzern genutzt wird. Unterschiede liegen in der Stabilität, Datenintegrität und Sicherheit, wobei Änderungen zunächst in der Testumgebung getestet und erst nach erfolgreichen Tests in die Produktivumgebung überführt werden sollten.

Sie erklären die Bedeutung von Regressionstests

Regressionstests sind wiederholte Tests, die nach Modifikationen an einem Programm durchgeführt werden, um sicherzustellen, dass die vorgenommenen Änderungen keine unbeabsichtigten Seiteneffekte oder neue Fehler verursacht haben. Ihr Ziel ist es zu überprüfen, dass Teile oder Features einer neuen Version, die unverändert bleiben sollen, tatsächlich weiterhin unverändert funktionieren.

Codereviews

Ziel von Code Review

- Qualität des Codes sichern -> Clean Code
- Code, der genau die Anforderungen erfüllt
- Wissensaustausch im Team
- Frühe Fehlererkennung

Code Review vorbereiten

- Checkliste für Anforderungen erstellen
- Checkliste für Code
    - Codierrichtlinien
    - Clean Code
    - SOLID

Code Review organisieren

- Gutachter bestimmen
    - Lieferung zustellen
    - Auftrag formulieren pro Gutachter basierend auf Checkliste
- Moderator bestimmen
- Termin und Raum organisieren

Review Kickoff

- Autor stellt Lieferung vor
- Gutachter lernen Lieferung und Kontext kennen
- Erstes Verständnis gewinnen
    - Aufgabenstellung verstehen
    - Anforderungen verstehen
- Architektur verstehen
    - Klassen und deren Verantwortlichkeiten
    - Kritische Stellen erkennen
- Tests ausführen
    - Tieferes Verständnis gewinnen

Review individuell vorbereiten

- Gutachter bearbeiten ihren Auftrag und Checkliste
    - Notieren ihre Befunde, möglichst konkret mit Bezug Checkliste und File (Klasse) / Codezeilennummer

Review durchführen

- Moderator nimmt Vorbereitungszeit der Gutachter auf, sind Gutachter nicht vorbereitet, wird Review abgebrochen
- Unter Leitung des Moderators werden die Befunde gesammelt und in die Befundliste aufgenommen, mit Referenz auf Checkliste und File/Codezeile.
- Es werden keine Lösungen diskutiert.
- Die Befunde werden in der Befundliste bewertet
- Das Reviewteam gibt in der Review Zusammenfassung eine Empfehlung ab basierend auf den Befunden
    - Review akzeptiert
        - Nur Nebenfehler, mit geringfügigen Änderungen
        - Überarbeitung notwendig, aber kein weiteres Review nötig
    - Review nicht akzeptiert,
        - Review muss nach Überarbeitung wiederholt werden.

Review Nachbearbeitung

- Projektleiter organisiert die Nachbearbeitung des Reviews
    - Formuliert Massnahmen pro Befund
    - Überwacht die Durchführung der Massnahmen

Bewertung der Nachbearbeitung

- Gutachter überprüfen die Massnahmen